UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/500,197	06/24/2004	Takashi Mita	30391-17	9239

23562 7590 05/06/2008

BAKER & MCKENZIE LLP
PATENT DEPARTMENT
2001 ROSS AVENUE
SUITE 2300
DALLAS, TX 75201

EXAMINER	
FENNEMA, ROBERT E	

ART UNIT	PAPER NUMBER
2183	

MAIL DATE	DELIVERY MODE
05/06/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/500,197	MITA ET AL.	
	Examiner	Art Unit	
	ROBERT E. FENNEMA	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 January 2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *3,4,6,8,10,13,14,16,18 and 20-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *3-4, 6, 8, 10, 13-14, 16, 18, and 20-36* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 3-4, 6, 8, 10, 13-14. 16, 18, and 20-36 have been considered. Claims 3-4, 6, 8, 10, 13-14, 16, 18, and 20-22 amended as per Applicant's request. Claims 23-36 added as per Applicant's request.

Claim Objections

2. In Claim 22, in the last line, the phrase "one of said" is repeated before "data registers".

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 25-26, 28-29, 32-33 and 35-36 are rejected under 35 U.S.C. 102(b) as being anticipated by Trimberger et al. (USPN 5,646,545, herein Trimberger).

5. As per Claim 25, Trimberger teaches: A computing system comprising:

a data storage (Column 22, Lines 6-8, the memory slices);

a loader (Column 17, Lines 41-47, a loader is required); and

a logic computing device (Column 1, Line 67, the PLD);

wherein said data storage stores a plurality of configuration data modules each of which includes data for forming a look-up table (Column 6, Lines 59-62),

said loader loads said configuration data modules from said data storage to said logic computing device in response to a plurality of load commands received from said logic computing device (Column 17, Lines 41-47),

said logic computing device comprises a selector (Column 7, Lines 33-36), a plurality of data memories (Column 2, Lines 15-17), a logic block (Column 1, Line 67 – Column 2, Line 5, a CLB), a routine detector (Column 26, Line 60 – Column 27, Line 14, a detector would be required to make the call), a parameter register (Column 2, Lines 22-23 and Column 7, Lines 23-26), a parameter buffer (Column 2, Lines 22-23 and Column 7, Lines 23-26) and a controller (Column 18, Lines 6-9, a sequencer),

wherein each of said data memories stores one of said configuration data modules loaded by said loader (Column 2, Lines 15-17),

said selector selects one data memory of said data memories in accordance with a control signal received from said controller (Column 7, Lines 33-36),

said logic block comprises at least one gate circuit(s) and at least one flip flop(s) and provides a logical function value of logic input data to outside said logic computing device as logic output data (Figure 3, 302-304 are gate circuits, 322-323 are flip flops),

said routine detector identifies a relationship between one configuration data module stored in said one data memory and another configuration data module (Column 17, Lines 28-34, it can differentiate between configurations),

said parameter register stores at least part of parameters stored in said flip flop(s) when said selector changes selection of said one data memory (Column 2, Lines 22-23 and Column 7, Lines 23-26),

said parameter buffer stores at least part of parameters stored in said flip flop(s) when said one configuration data module stored in said one data memory calls another configuration data module as a subroutine (Column 2, Lines 22-23 and Column 7, Lines 23-26), and

said controller controls logic computing in said logic block (Column 7, Lines 23-26, as the configuration changes due to a call/return, the parameters are clocked into the register, and a controller must be present to direct the register to do this).

6. As per Claim 26, Trimberger teaches: The computing system according to claim 25, wherein said controller restores the parameters stored in said parameter buffer to said flip flop(s) when one configuration data module stored in said one data memory calls back to another configuration data module as a main routine (Column 22, Lines 10-14).

7. As per Claim 28, Trimberger teaches: The computing system according to claim 25, further comprising a compiler which creates each of said configuration data modules based on each of a plurality of source program modules (Examiner is taking official

notice that some kind of compiler is required to program an FPGA (or any other kind of programmable logic hardware) or run any kind of source program on a computer).

8. As per Claim 29, Trimberger teaches: The computing system according to claim 25, wherein:

said logic computing device generates a load command in the process of computing by said logic block (Column 17, Lines 52-55); and

said loader loads the configuration data modules, indicated by the load command in the process, from said data storage to one of said data memories (Column 17, Lines 31-33 and 52-58, the loader loads configuration data, and the FPGA initiates reconfiguration of CLB's as necessary).

9. As per Claim 32, Trimberger teaches: A computing method comprising:

storing a plurality of configuration data modules in a data storage (Column 22, Lines 6-8, the memory slices), each of said configuration data modules includes data for forming a look-up table (Column 6, Lines 59-62);

loading, by a loader, said configuration data modules from said data storage to a logic computing device in response to a plurality of load commands received from said logic computing device (Column 17, Lines 41-47);

storing each of said configuration data modules loaded by said loader in each of a plurality of data memories of said logic computing device (Column 17, Lines 41-47);

selecting, by a selector of said logic computing device, one data memory of said data memories in accordance with a control signal received from a controller of said logic computing device (Column 7, Lines 33-36);

configuring a logic block of said logic computing device so as to comprise at least one gate circuit(s) and at least one flip flop(s) (Figure 3, 302-304 are gate circuits, 322-323 are flip flops);

providing, by said logic block, a logical function value of logic input data to outside said logic computing device as logic output data (Figure 3, this is what CLBs do);

identifying, by a routine detector of said logic computing device, a relationship between one configuration data module stored in said one data memory and another configuration data module (Column 17, Lines 28-34, it can differentiate between configurations);

storing at least part of parameters, stored in said flip flop(s), in a parameter register of said logic computing device when said selector changes selection of said one data memory (Column 2, Lines 22-23 and Column 7, Lines 23-26);

storing at least part of parameters, stored in said flip flop(s), in a parameter buffer of said logic computing device when said one configuration data module stored in said one data memory calls another configuration data module as a subroutine (Column 2, Lines 22-23 and Column 7, Lines 23-26); and

controlling, by said controller, logic computing in said logic block (Column 7, Lines 23-26, as the configuration changes due to a call/return, the parameters are clocked into the register, and a controller must be present to direct the register to do this).

10. As per Claim 33, Trimberger teaches: The computing method according to claim 32, further comprising restoring, by said controller, the parameters stored in said parameter buffer to said flip flop(s) when one configuration data module stored in said one data memory calls back to another configuration data module as a main routine (Column 22, Lines 10-14).

11. As per Claim 35, Trimberger teaches: The computing method according to claim 32, further comprising creating, by a compiler, each of said configuration data modules based on each of a plurality of source program modules (Examiner is taking official notice that some kind of compiler is required to program an FPGA (or any other kind of programmable logic hardware) or run any kind of source program on a computer).

12. As per Claim 36, Trimberger teaches: The computing method according to claim 32, further comprising:

generating a load command in the process of computing by said logic block (Column 17, Lines 52-55); and

loading, by said loader, the configuration data module, indicated by the load command in the process, from said data storage to one of said data memories (Column 17, Lines 31-33 and 52-58, the loader loads configuration data, and the FPGA initiates reconfiguration of CLB's as necessary).

13. Claims 6, 10, 16, 21-23, and 30-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Trimberger, in view of Butts et al. (USPN 5,036,473, herein Butts).

14. As per Claim 23, Trimberger teaches: A computing system comprising:

a data storage (Column 22, Lines 6-8, the memory slices);

a loader (Column 17, Lines 41-47, a loader is required); and

a logic computing device (Column 1, Line 67, the PLD),

wherein said data storage stores a plurality of configuration data modules each of which includes data for forming a look-up table (Column 6, Lines 59-62),

said loader loads said configuration data modules from said data storage to said logic computing device in response to a plurality of load commands (Column 17, Lines 41-47),

said logic computing device comprises, a logic block (Column 1, Line 67 – Column 2, Line 5, a CLB), a routine detector (Column 26, Line 60 – Column 27, Line 14, a detector would be required to make the call), a parameter register (Column 2, Lines

22-23 and Column 7, Lines 23-26), a parameter buffer (Column 2, Lines 22-23 and Column 7, Lines 23-26) and a controller (Column 18, Lines 6-9, a sequencer),

said logic block comprises at least one gate circuit(s) and at least one flip flop(s), and provides a logical function value of logic input data to outside said logic computing device as logic output data (Figure 3, 302-304 are gate circuits, 322-323 are flip flops),

said routine detector identifies a relationship between one configuration data module stored in one data register of said data registers and another configuration data module (Column 17, Lines 28-34, it can differentiate between configurations),

said parameter register storages at least part of parameters stored in said flip flop(s) when said configuration data modules stored in said shift register are shifted (Column 2, Lines 22-23 and Column 7, Lines 23-26),

said parameter buffer stores at least part of parameters stored in said flip flop(s) when one configuration data modules stored in said one data register calls another configuration data module as a subroutine (Column 2, Lines 22-23 and Column 7, Lines 23-26), and

said controller controls logic computing in said logic block (Column 7, Lines 23-26, as the configuration changes due to a call/return, the parameters are clocked into the register, and a controller must be present to direct the register to do this), but fails to teach:

a shift register;

wherein said shift register includes a plurality of data registers each of which stores one of said configuration data modules loaded by said loader.

While Trimberger teaches storing configuration data in a series of memory slices, Trimberger is silent towards the memory slices being arranged as shift registers, nor exactly how the data from off-chip is sent to the memory. However, Butts teaches that in reconfigurable systems (with an example being the LCA chip), the reconfigurable features are controlled by shifting in configuration data (Column 7, Line 67 – Column 8, Line 2). Given this disclosure, one of ordinary skill in the art would have used Butts specific teachings of how to program the reconfigurable portions given the lack of specifics in Trimberger, and made the memory "shift registers", in order to accept the shifted data.

15. As per Claim 21, Trimberger teaches: The computing system according to claim 23, wherein:

said logic computing device generates a load command in the process of computing by said logic block (Column 17, Lines 52-55); and

said loader loads the configuration data module indicated by the load command in the process, from said data storage to one of said data registers (Column 17, Lines 31-33 and 52-58, the loader loads configuration data, and the FPGA initiates reconfiguration of CLB's as necessary).

16. As per Claim 24, Trimberger teaches: The computing system according to claim 23, wherein said controller returns the parameters stored in said parameter buffer to

said logic block as input of said gate circuit(s) when said another configuration data module as the subroutine is shifted to said one data register (Column 22, Lines 10-14).

17. As per Claim 6, Trimberger teaches: The logic computing system according to claim 23, wherein said controller:

restores the parameters stored in said parameter buffer to said flip flop(s) when one configuration data module stored in said one data register calls back to another configuration data module as a main routine (Column 22, Lines 10-14).

18. As per Claim 10, Trimberger teaches: The computing system according to claim 23, comprising a compiler which creates each of said configuration data modules based on each of a plurality of source program modules (Examiner is taking official notice that some kind of compiler is required to program an FPGA (or any other kind of programmable logic hardware) or run any kind of source program on a computer). Claim 20 has substantially similar limitations to Claim 10 and is rejected for the same reasons.

19. As per Claim 30, Trimberger teaches: A computing method comprising:

storing a plurality of configuration data modules in a data storage (Column 22, Lines 6-8, the memory slices), each of said configuration data modules includes data for forming a look-up table (Column 6, Lines 59-62);

loading, by a loader, said configuration data modules from said data storage to a logic computing device in response to a plurality of load commands (Column 17, Lines 41-47);

storing each of said configuration data modules loaded by said loader in each of a plurality of data registers (Column 17, Lines 41-47);

configuring a logic block of said logic computing device so as to comprise at least one gate circuit(s) and at least one flip flop(s) (Figure 3, 302-304 are gate circuits, 322-323 are flip flops);

providing, by said logic block, a logical function value of logic input data to outside said logic computing device as logic output data (Figure 3, this is what CLBs are designed to do);

identifying, by a routine detector of said logic computing device, a relationship between one configuration data module stored in one data register of said data registers and another configuration data module (Column 17, Lines 28-34, it can differentiate between configurations);

storing at least part of parameters, stored in said flip flop(s), in a parameter register of said logic computing device (Column 2, Lines 22-23 and Column 7, Lines 23-26);

storing at least part of parameters, stored in said flip flop(s), in a parameter buffer of said logic computing device when said one configuration data module stored in said

one data register calls another configuration data module as a subroutine (Column 2, Lines 22-23 and Column 7, Lines 23-26); and

controlling, by a controller of said logic computing device, logic computing in said logic block, but fails to teach:

a shifted register; and

storing in a parameter register when said shift register are shifted.

While Trimberger teaches storing configuration data in a series of memory slices, Trimberger is silent towards the memory slices being arranged as shift registers, nor exactly how the data from off-chip is sent to the memory. However, Butts teaches that in reconfigurable systems (with an example being the LCA chip), the reconfigurable features are controlled by shifting in configuration data (Column 7, Line 67 – Column 8, Line 2). Given this disclosure, one of ordinary skill in the art would have used Butts specific teachings of how to program the reconfigurable portions given the lack of specifics in Trimberger, and made the memory "shift registers", in order to accept the shifted data.

20. As per Claim 22, Trimberger teaches: The logic computing method according to claim 30, further comprising:

generating a load command in the process of computing by said logic block (Column 17, Lines 52-55); and

loading by said loader the configuration data module indicated by the load command in the process, from said data storage to one of said one of said data registers (Column 17, Lines 31-33 and 52-58, the loader loads configuration data, and the FPGA initiates reconfiguration of CLB's as necessary).

21. As per Claim 31, Trimberger teaches: The computing method according to claim 30, further comprising returning, by said controller, the parameters stored in said parameter buffer to said logic block as input of said gate circuit(s), when said another configuration data module as the subroutine is shifted to said one data register (Column 22, Lines 10-14).

22. As per Claim 16, Trimberger teaches: The computing method according to claim 30, comprising:

restoring, by said controller, the parameters stored in said parameter buffer to said flip flop(s) when one configuration data module stored in said one data register calls back to another configuration data module as a main routine (Column 22, Lines 10-14).

23. Claims 4 and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Trimberger, further in view of Liu et al. (herein Liu).

24. As per Claim 4, Trimberger teaches: The logic computing system according to claim 23, wherein:

said logic block refers to said one configuration data module stored in said one data memory (Column 18, Lines 12-26), but fails to teach:

said selector changes selection of said one data memory with another data memory among said data memories circularly.

Trimberger has taught that the data storage units containing configuration data modules (the memory slices) are arranged as shift registers as shown in the previous claims, but have not taught that these shift registers are arranged in a circular fashion. However, Liu teaches a method for partitioning a system similar to Trimberger's, which also offers an advantage of outperforming the force-directed scheduling method (Abstract) used by Trimberger (Trimberger, Column 30, Lines 19-23). As can be seen by Figure 1, the configurable logic is laid out as a circular shift register. Given the advantage of increased performance over the scheduling method employed by Trimberger, one of ordinary skill in the art at the time the invention was made would have made use of Liu's invention, which would have also made the shift registers need to be laid out in a circular fashion.

Claim 14 is substantially similar to Claim 4, and is rejected for the same reasons.

25. Claims 3 and 13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Trimberger and Butts, further in view of Liu et al. (herein Liu).

26. As per Claim 3, Trimberger and Butts teaches: The logic computing system according to claim 23, wherein:

said logic block refers only to said one configuration data module stored in said one data register (Column 7, Lines 23-26, only one configuration is active at any time), but fails to teach:

said shift register shifts said configuration data modules among said data registers circularly.

Trimberger and Butts have taught that the data storage units containing configuration data modules (the memory slices) are arranged as shift registers as shown in the previous claims, but have not taught that these shift registers are arranged in a circular fashion. However, Liu teaches a method for partitioning a system similar to Trimberger's, which also offers an advantage of outperforming the force-directed scheduling method (Abstract) used by Trimberger (Trimberger, Column 30, Lines 19-23). As can be seen by Figure 1, the configurable logic is laid out as a circular shift register. Given the advantage of increased performance over the scheduling method employed by Trimberger, one of ordinary skill in the art at the time the invention was made would have made use of Liu's invention, which would have also made the shift registers need to be laid out in a circular fashion.

Claim 13 is substantially similar to claim 3 and is rejected for the same reasons.

27. Claims 8, 27, and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Trimberger, in view of Patterson et al. (herein Patterson).

28. As per Claim 8, Trimberger teaches: The logic computing system according to claim 23, wherein:

said routine detector detects a call by said one configuration data module to said another configuration data modules as the subroutine (Column 26, Line 60 – Column 27, Line 14), but fails to teach:

said controller searches said data registers for said another configuration data module as the subroutine when said routine detector detects the call, and sends a load command to said loader in a case where said another configuration data module as the subroutine is not searched out; and

said loaded loads said another configuration data module as the subroutine from said data storage to one of said data registers in response to the load command received from said controller.

Claims 27 and 34 are substantially similar to claim 8 and are rejected for the same reasons.

Trimberger has taught a detector which can detect calls, and a controller to tell a loader to load modules, but has not taught searching the plurality of data storage units for a configurable data module as specified by a call, and then loading that module if it was not found in a search. Trimberger teaches these things to account for the fact that there are not enough configurable logic elements in his system to handle the entire

program, so it is broken up and partitioned into pieces, which can be swapped in and out. Patterson has taught a method called paging which is used to allow a very large program (or programs), much bigger than main memory, to be used by a computer, by swapping pages in and out of memory as they are required, creating the illusion of a much larger memory space (Pages 439-441). Foldoc further describes paging and page faults, and is referred to as extrinsic evidence on the functionality of paging (see "paging" and "page fault" documents). The advantage of paging, as stated earlier, is virtually increasing the amount of memory that appears to be available in a system by swapping "pages" of memory in and out of main memory, in a very similar way to how Trimberger swaps configuration modules in and out of the CLB's. The difference is that in paging, a search is first conducted among the pages in memory, and the appropriate page is then loaded if not found (see "page fault" entry of Foldoc), which has the further obvious advantage of not having to reconfigure/fetch on every call. Given these advantages, it would have been obvious to one of ordinary skill in the art at the time the invention was made to fully incorporate the idea of paging into Trimberger's invention by searching for configurations before fully reloading the data storage units to maximize performance.

29. Claim 18 is rejected under 35 U.S.C. 103(a) as being unpatentable over Trimberger and Butts, in view of Patterson.

30. As per Claim 18, Trimberger teaches:

detecting, by said routine detector, a call by said one configuration data module to said another configuration data module as the subroutine (Column 26, Line 60 – Column 27, Line 14), but fails to teach:

searching, by said controller, said data registers for said another configuration data module as the subroutine when the call is detected by said routine detector;

sending, by said controller, a load command to said loader in a case where said another configuration data module as a subroutine is not searched out; and

loading, by said loader, said another configuration data module as the subroutine from said data storage to one of said data registers in response to the load command received from said controller.

Trimberger has taught a detector which can detect calls, and a controller to tell a loader to load modules, but has not taught searching the plurality of data storage units for a configurable data module as specified by a call, and then loading that module if it was not found in a search. Trimberger teaches these things to account for the fact that there are not enough configurable logic elements in his system to handle the entire program, so it is broken up and partitioned into pieces, which can be swapped in and out. Patterson has taught a method called paging which is used to allow a very large program (or programs), much bigger than main memory, to be used by a computer, by swapping pages in and out of memory as they are required, creating the illusion of a much larger memory space (Pages 439-441). Foldoc further describes paging and page faults, and is referred to as extrinsic evidence on the functionality of paging (see "paging" and "page fault" documents). The advantage of paging, as stated earlier, is

virtually increasing the amount of memory that appears to be available in a system by swapping "pages" of memory in and out of main memory, in a very similar way to how Trimberger swaps configuration modules in and out of the CLB's. The difference is that in paging, a search is first conducted among the pages in memory, and the appropriate page is then loaded if not found (see "page fault" entry of Foldoc), which has the further obvious advantage of not having to reconfigure/fetch on every call. Given these advantages, it would have been obvious to one of ordinary skill in the art at the time the invention was made to fully incorporate the idea of paging into Trimberger's invention by searching for configurations before fully reloading the data storage units to maximize performance.

Response to Arguments

31. Regarding Applicants arguments towards Claims 23 and 30, Examiner agrees that Trimberger alone does not teach the claims, and has provided rejections under 103 for these claims.

32. Regarding Applicants arguments in regards to Claims 3 and 13, Applicant has argued that Liu indicates that the same FPGA is temporally shared by all of the logic in different stages, and because of those, the logic block cannot refer to only one configuration data module stored in one data register. However, Examiner disagrees that this means that Liu does not teach referring to only one configuration data module. At any given time, only one configuration is being used (each micro-cycle only refers to

a single configuration), therefore Examiner believes it is fair to say that the combination does teach the limitation, as there is no restriction in the claim as to how long the FPGA/logic block must refer to the one configuration.

33. Regarding Applicants arguments towards Claims 25 and 32, Applicant has argued that Trimberger does not teach or suggest the loader and the logic computing device reciting in the claims. Examiner disagrees with this assertion, and notes Column 17, Lines 41-47, which shows that loading is taking place, and Examiner asserts that something is doing the loading, and that is the loader. While Trimberger does not explicitly lay out what module is specifically the loader, a loader must exist to perform the functionality described in Column 17. Additionally, Examiner has mapped the PLD to the logic computing device referred to in these claims.

Conclusion

34. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the

shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to ROBERT E. FENNEMA whose telephone number is (571)272-2748. The examiner can normally be reached on Monday-Friday, 8:30-6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Eddie P Chan/
Supervisory Patent Examiner, Art Unit 2183

Robert E Fennema
Examiner
Art Unit 2183

RF